Core Java:
Introduction to Java

Persistent Interactive | Persistent University

Key learning points

At the end of this module, you will be able to:

- Understand why we need Java
- Understand the History of Java
- Get to know the Importance of Java to internet
- Java platform and libraries
- Compilation and execution of Java programs
- Understand the features of Java
- To know different versions of Java
- Write your first Java program using Eclipse IDE

Configurations required

- JDK 1.11
- Eclipse Luna
- MySQL
- Java 11 Documentation

Why Java?

Crunch Numbers

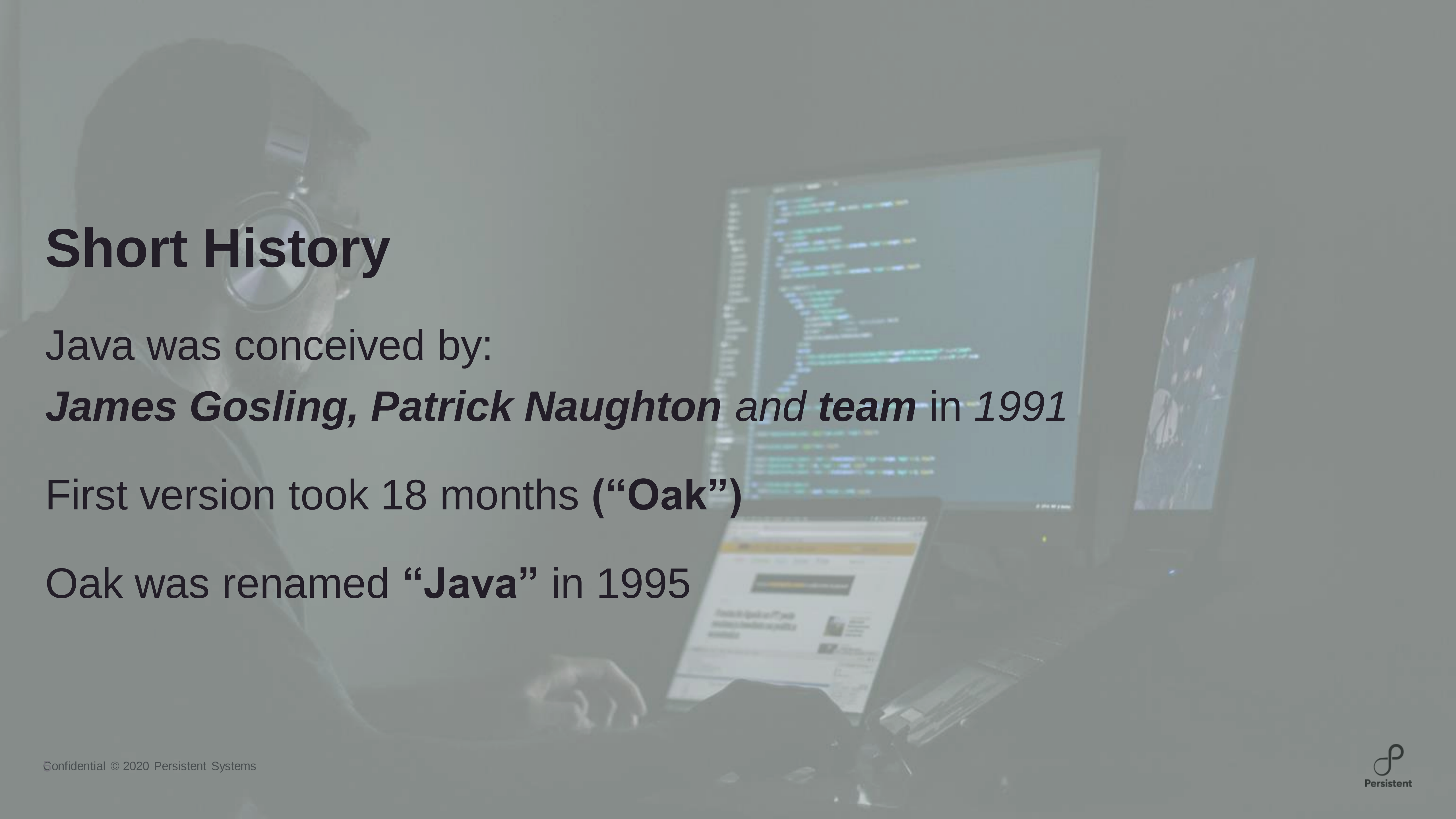
Play Games

Process Words

Store Data

Mission-Critical apps

Trading/Financial apps



Short History

Java was conceived by:

James Gosling, Patrick Naughton *and* ***team*** in *1991*

First version took 18 months **(“Oak”)**

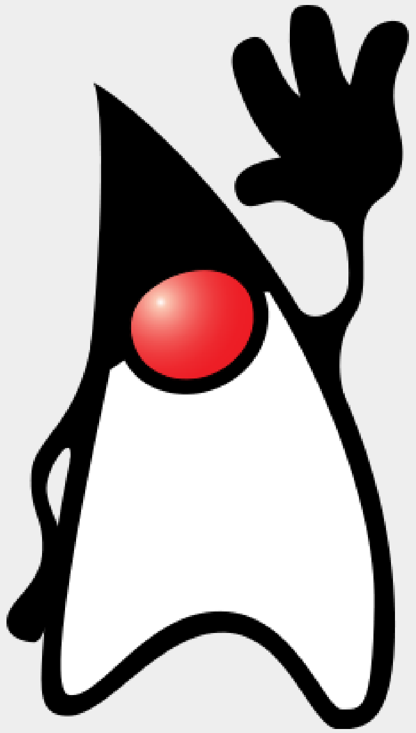
Oak was renamed **“Java”** in 1995

Why is Java important to the internet?

Security

Portability

Java platform and class libraries

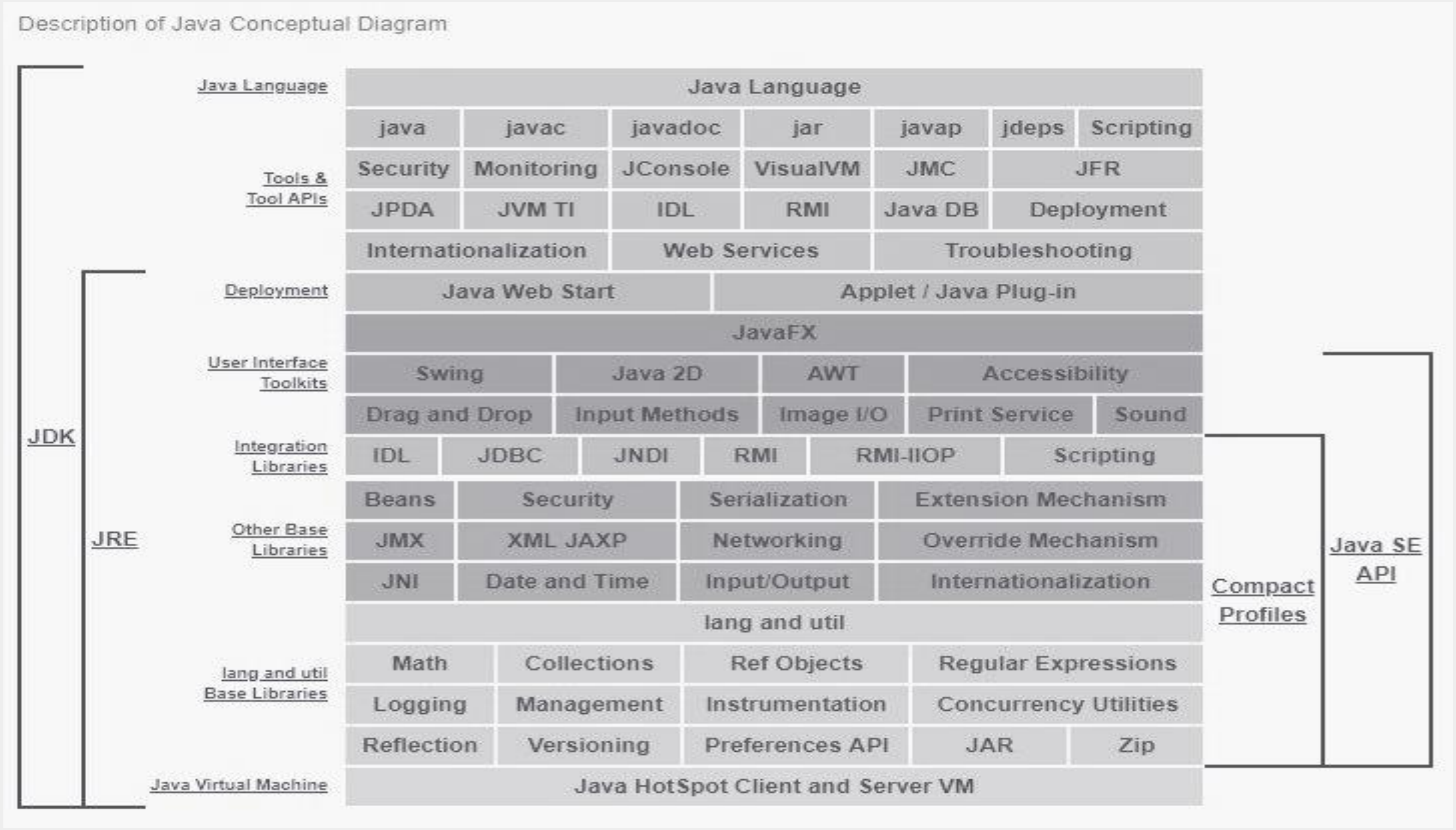


Java Platform
Standard Edition (Java SE)

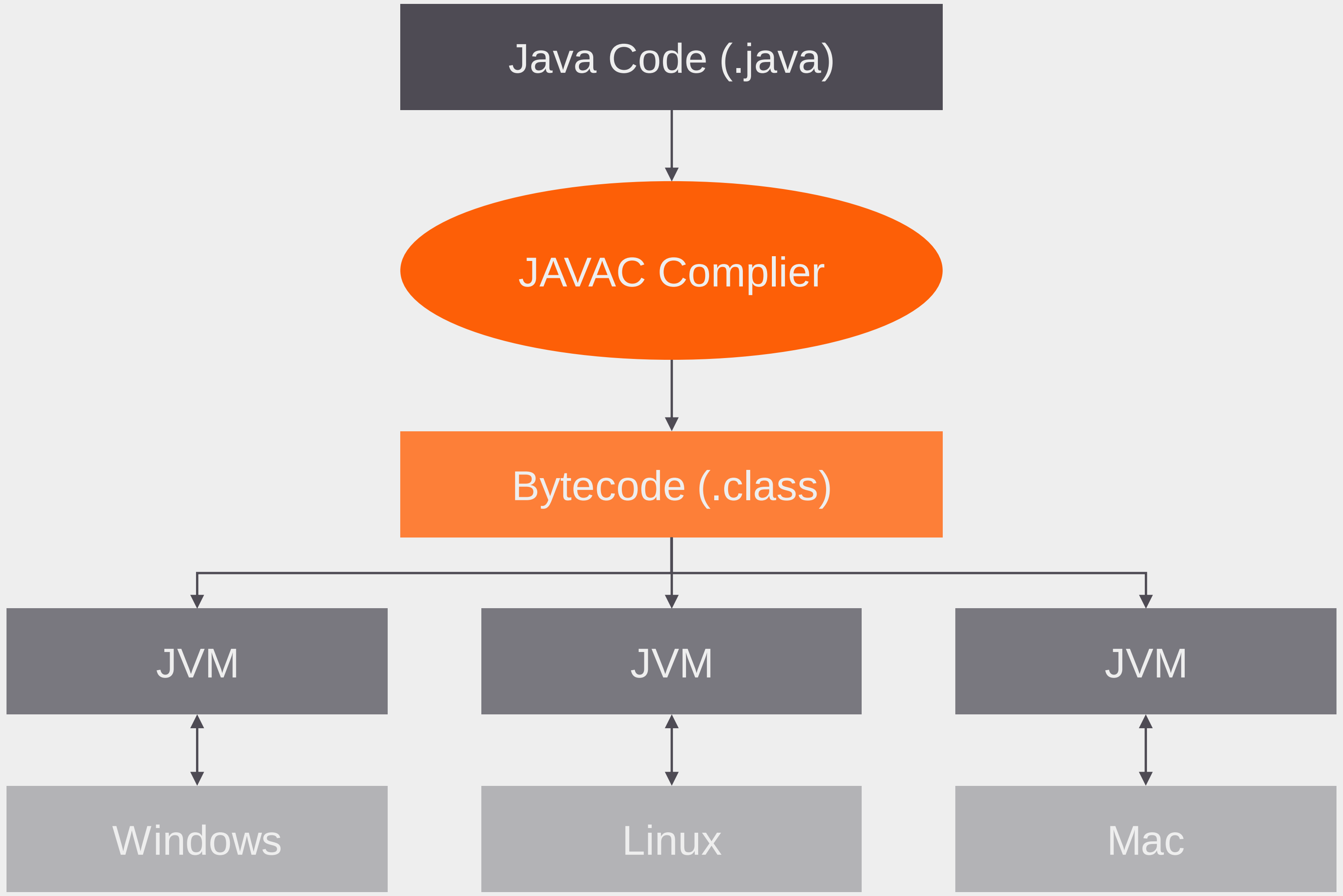
Java Platform
Enterprise Edition (Java EE)

Java Platform
Micro Edition (Java ME)

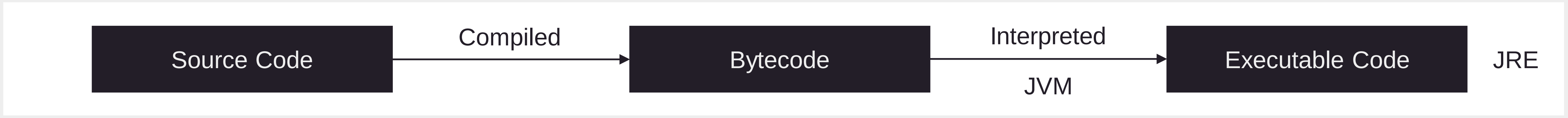
Java platform and class libraries

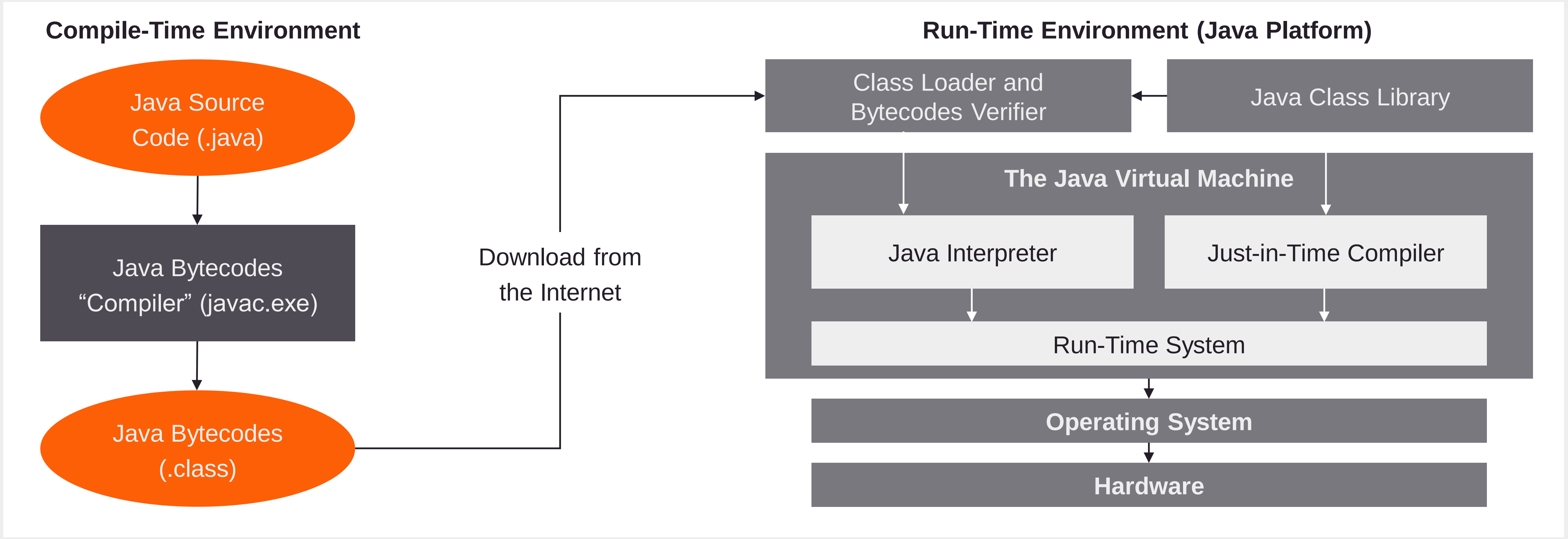


The Bytecode



Compilation and execution





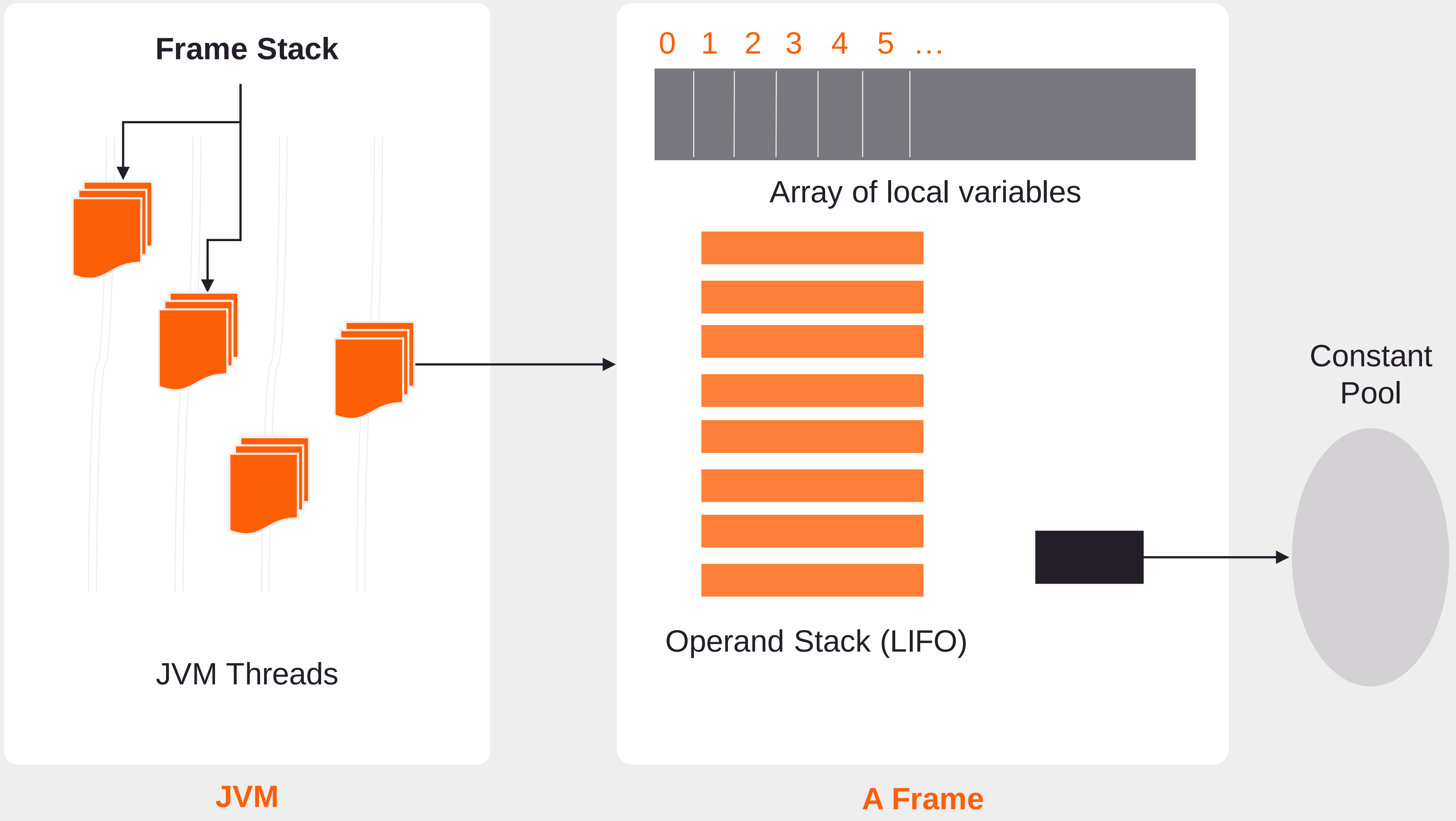
The Bytecode

- Bytecode is the intermediate representation of Java programs
- Bytecode understanding helps in debugging and doing performance and memory usage tuning

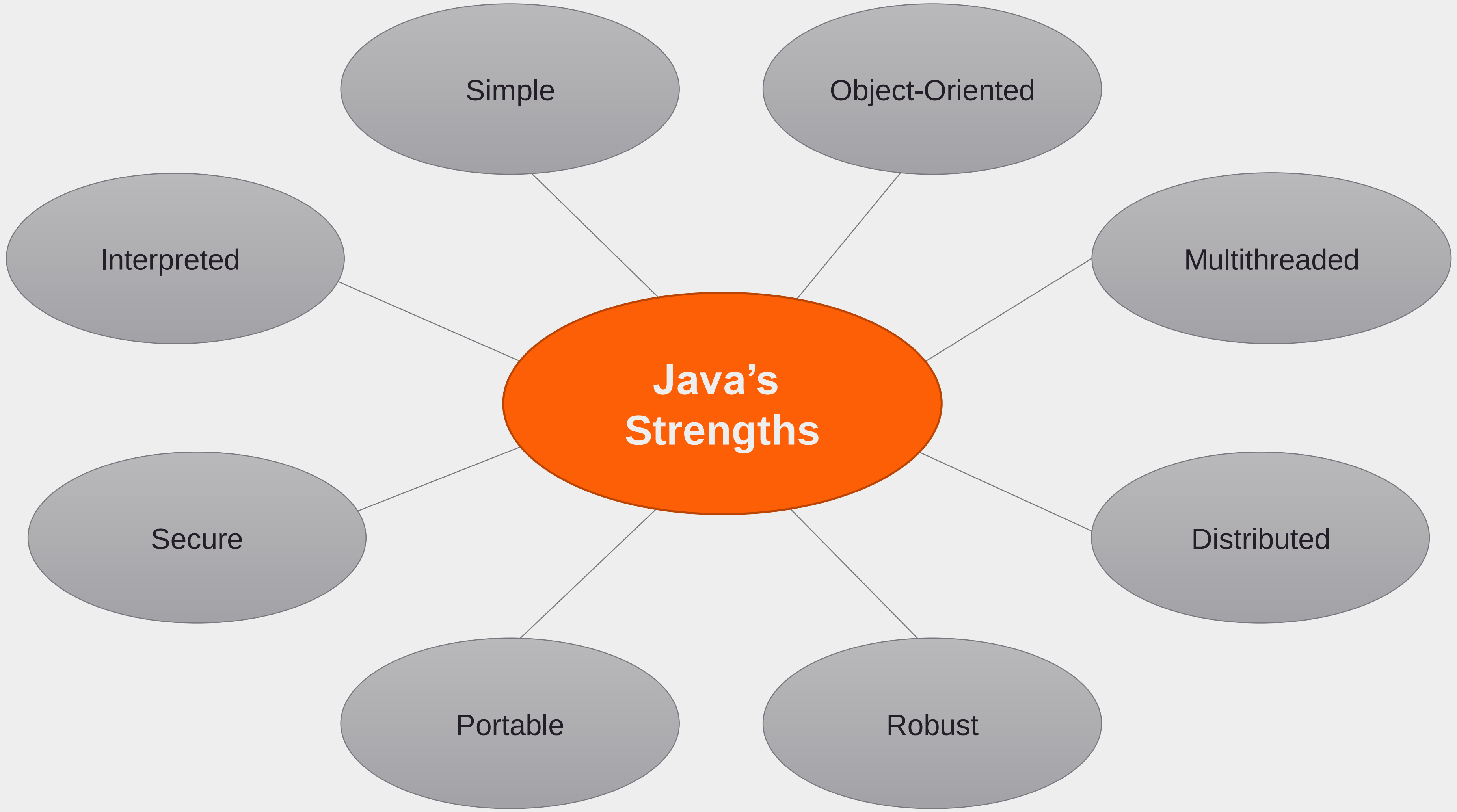
Java Virtual Machine

- JVM uses stack-based model of computation
- Each thread has a JVM stack which stores frames
- Each time a method is invoked a new stack frame is created
- Each stack frame consists of Operand Stack, Array of local variables, and a reference to Constant Pool

Java Virtual Machine



Why Java?


Java's Strengths
Simple
Object-Oriented
Interpreted
Multithreaded
Secure
Distributed
Portable
Robust

Features added in Java versions

Jdk 1.1

- JDBC (Java Database Connectivity)
- Inner Classes
- Java Beans
- RMI (Remote Method Invocation)
- Reflection (introspection only)

Jdk 1.2

- Collections framework
- Java String memory map for constants
- Just In Time (JIT) compiler
- Jar Signer for signing Java Archive (JAR) files
- Policy Tool for granting access to system resources
- Java Foundation Classes (JFC) which consists of Swing 1.0, Drag and Drop, and Java 2D class libraries
- Java Plug-in
- Scrollable result sets, BLOB, CLOB, batch update, user-defined types in JDBC
- Audio support in Applets

Jdk 1.3

- Java sound
- Jar indexing

Jdk 1.4

- XML Processing
- Java Print Service
- Logging API
- Java Web Start
- JDBC 3.0 API
- Assertions
- Preferences API
- Chained Exception
- IPv6 Support
- Regular Expressions
- Image I/O API

Features added in Java versions

Java 5

- Generics
- Enhanced for Loop
- Autoboxing/Unboxing
- Typesafe Enums
- Varargs
- Static Import
- Metadata (Annotations)

Java 6

- Scripting Language Support
- Collection API interfaces like Deque, NavigableSet, NavigableMap
- Collection API classes as ArrayDeque
- JDBC 4.0 API

Java 7

- Strings in Switch Statement
- Type Inference for Generic Instance Creation
- Multiple Exception Handling
- Try with Resources
- Binary Literals, underscore in literals
- Diamond Syntax

Java 8

- Lambda Expressions
- Pipelines and Streams
- Date and Time API
- Default Methods and Static methods in interfaces
- Type Annotations
- Functional Interfaces
- Parallel Operations

Features added in Java versions

Java 9

- Factory Methods for Immutable List, Set, Map and Map.Entry
- Private Methods in Interfaces
- Reactive Streams
- Diamond Operator for Anonymous Inner Class
- Optional Class Improvements
- Stream API Improvements
- Enhanced @Deprecated Annotation

Java 10

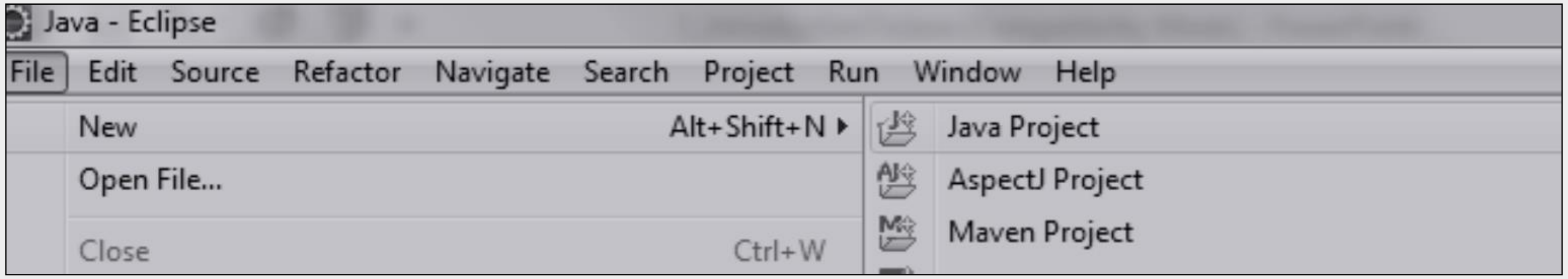
- Local-Variable Type Inference
- Collection API copyOf(Collection) Method
- Optional API orElseThrow() method

Java 11

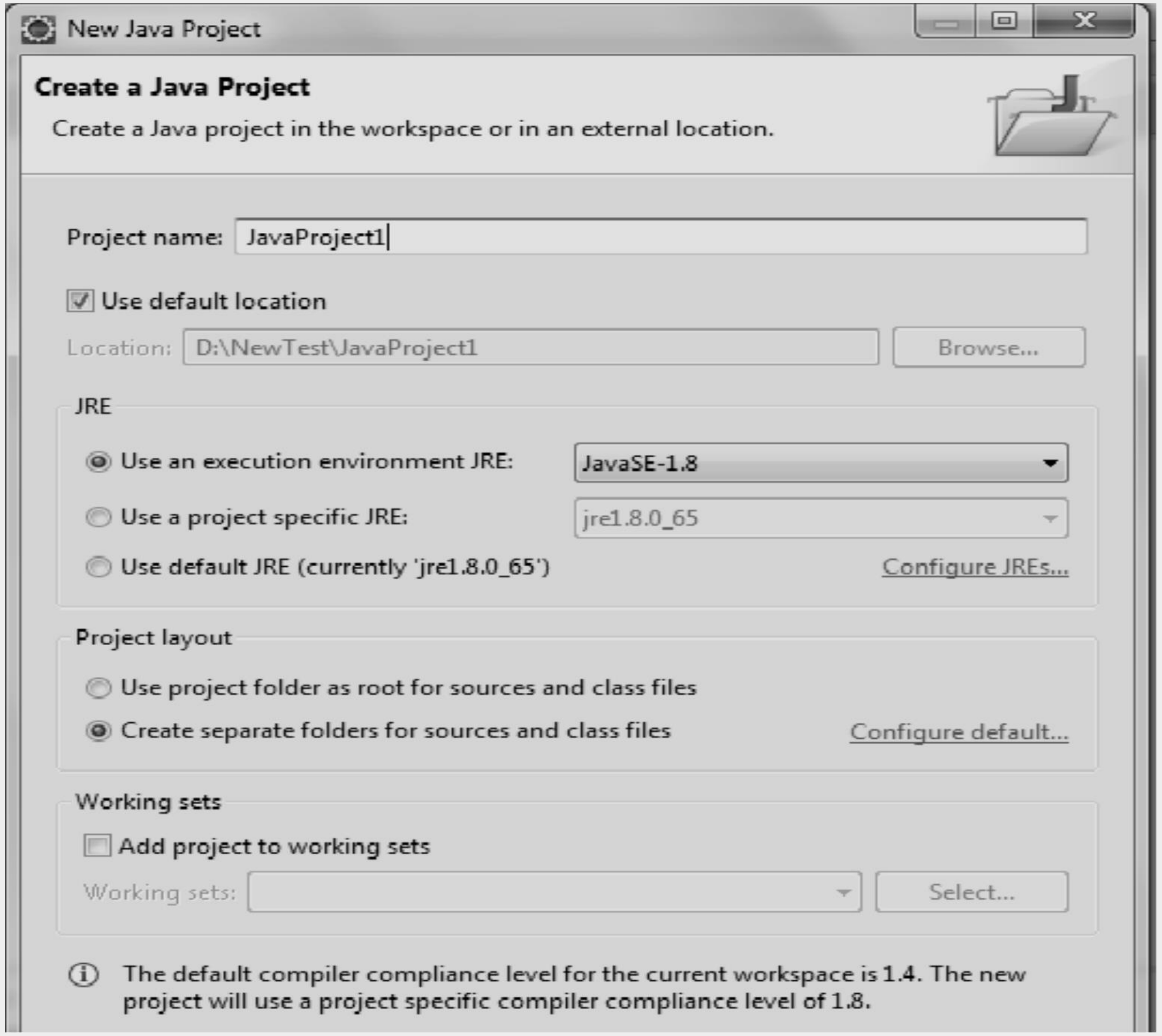
- Running Java File with single command
- New utility methods in String class
- Local-Variable Syntax for Lambda Parameters
- Nested Based Access Control
- JEP 321: HTTP Client
- Reading/Writing Strings to and from the Files

How to develop application using eclipse?

- Open Eclipse
- Specify the workspace
- Create a new Java Project
  - File -> New -> Java Project



Specify project name and click finish


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: JavaProject1

[✓] Use default location

Location: D:\NewTest\JavaProject1 Browse...

JRE

(●) Use an execution environment JRE: JavaSE-1.8

( ) Use a project specific JRE: jre1.8.0_65

( ) Use default JRE (currently 'jre1.8.0_65') Configure JREs...

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files Configure default...

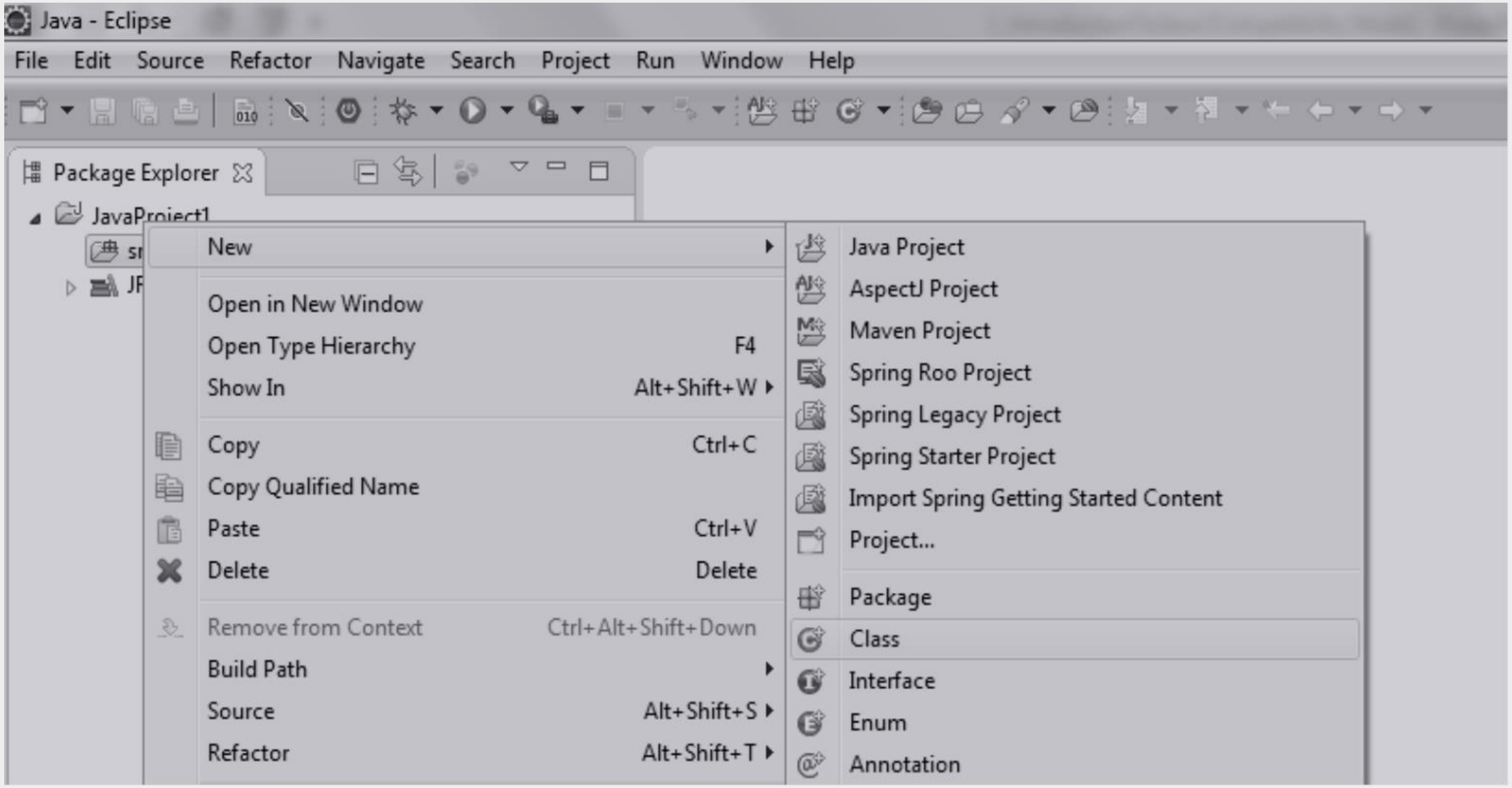
Working sets

[ ] Add project to working sets

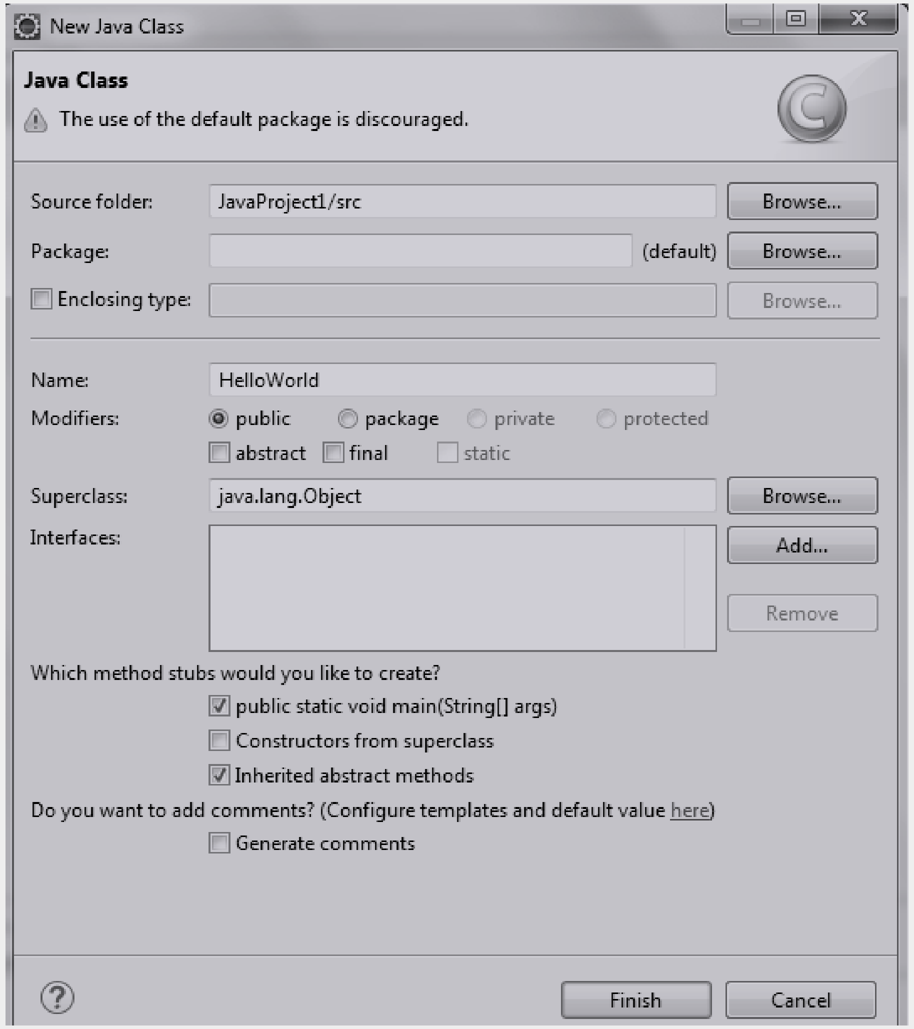
Working sets: Select...

The default compiler compliance level for the current workspace is 1.4. The new project will use a project specific compiler compliance level of 1.8.

Create a new class



Specify class name



Java Class

The use of the default package is discouraged.

Source folder:	JavaProject1/src	Browse...
Package:	(default)	Browse...
☐ Enclosing type:		Browse...
Name:	HelloWorld	
Modifiers:	[● public] ○ package ○ private ○ protected ☐ abstract ☐ final ☐ static	
Superclass:	java.lang.Object	Browse...
Interfaces:		Add... Remove

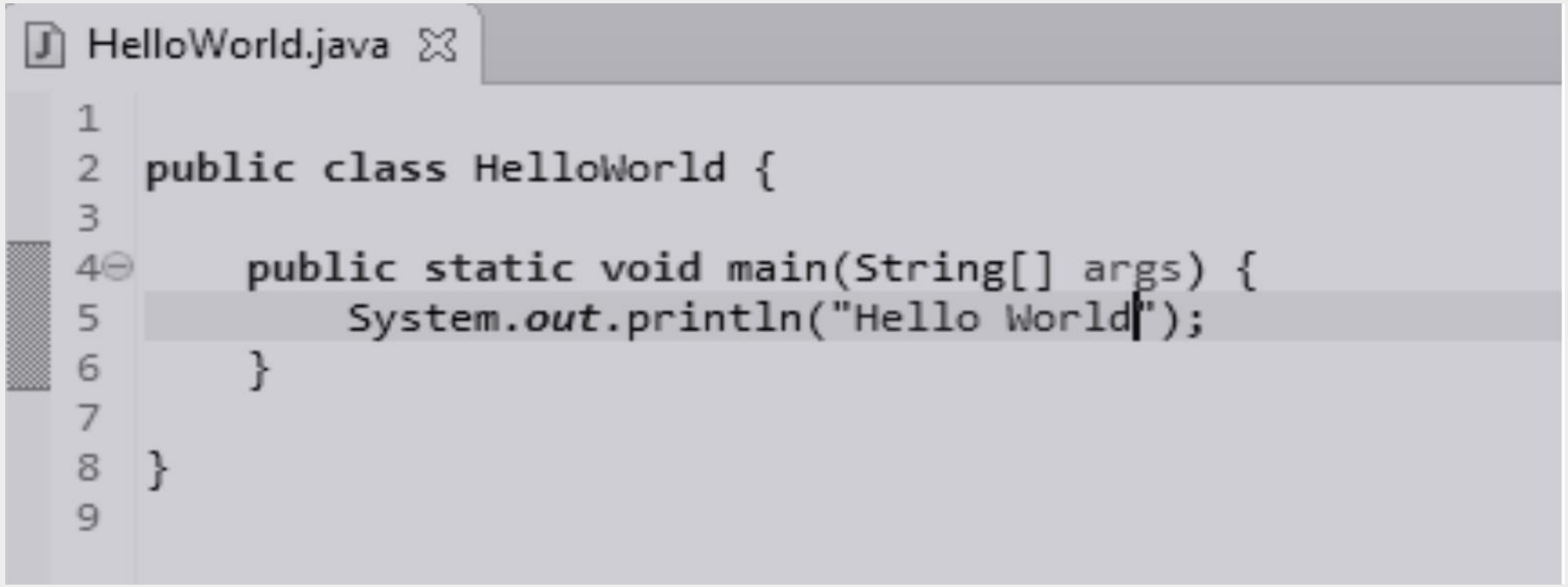
Which method stubs would you like to create?

- ☑ public static void main(String[] args)
- ☐ Constructors from superclass
- ☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

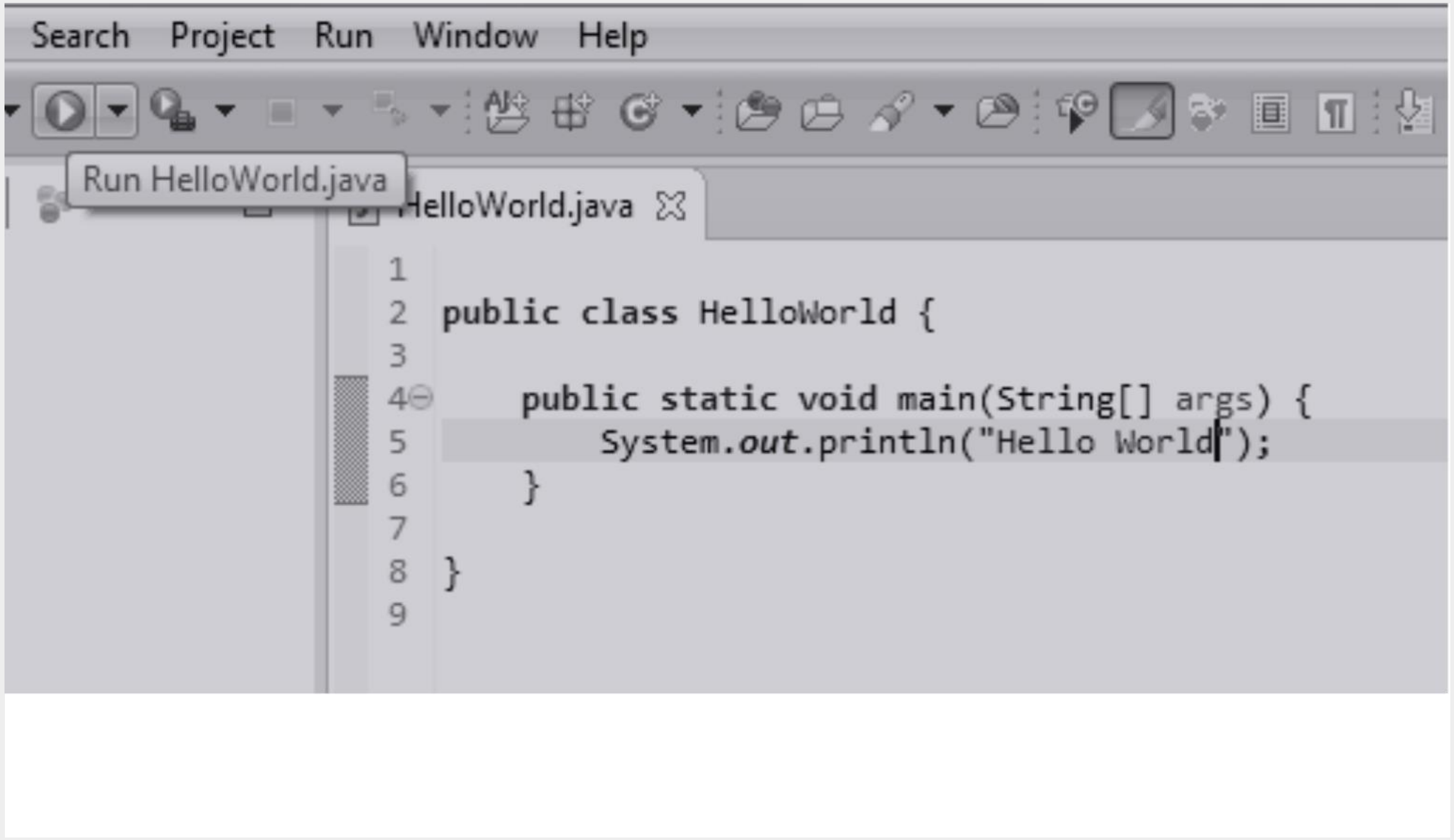
- ☐ Generate comments

Write the following code


HelloWorld.java

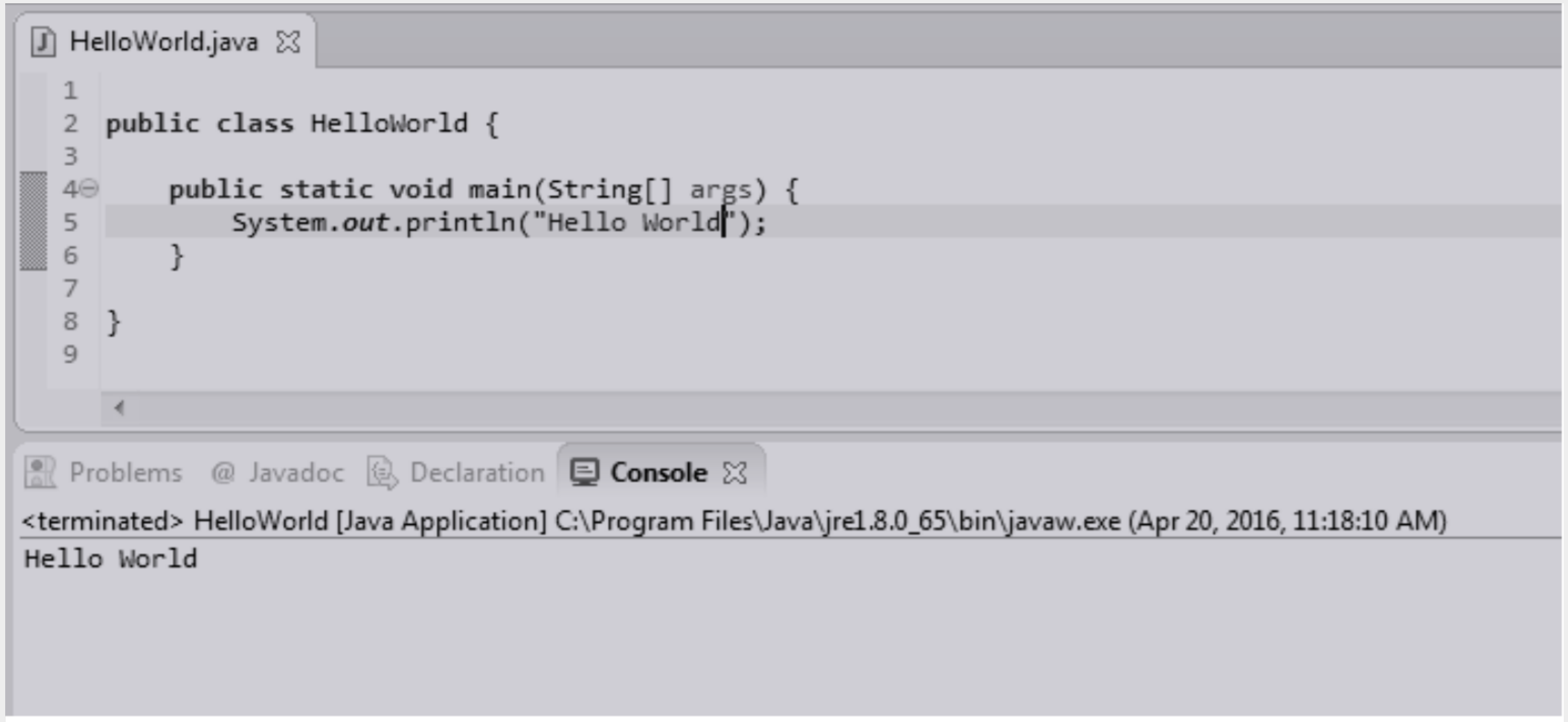
```

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}

```

Save and execute



```

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}

```

Observe output on the console



```

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}

```

```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Apr 20, 2016, 11:18:10 AM)
Hello World
```

Summary: Session

With this we have come to the end of our first session where we discussed:

- Basics of Java
- History, features and various versions of Java
- How to develop Java application using Eclipse

At the end of this session, we expect you to:

- Understand the basics of Java
- Develop an application using Java

Appendix

References

Thank you

Reference material: Websites & blogs

- https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html
- https://java.com/en/download/faq/whatis_java.xml
- https://docs.oracle.com/cd/E19455-01/806-3461/6jck06gqd/index.html
- http://javarevisited.blogspot.in/2011/12/jre-jvm-jdk-jit-in-java-programming.html
- https://docs.oracle.com/javase/tutorial/java/concepts/
- http://www.javatpoint.com/java-oops-concepts

Reference material: Books

Head First Java

- By: Kathy Sierra, Bert Bates
- Publisher: O’Reilly Media, Inc.

Java Complete Reference

- By: Herbert Schildt

Persistent

Thank you!

Persistent Interactive | Persistent University